任务1. 声明一个CPU类，包含等级（rank）、频率（frequency）、电压（voltage）等属性，有两个公有成员函数run、stop。其中，rank为枚举类型CPU_Rank，声明为enum CPU_Rank{P1=1，P2，P3，P4，P5，P6，P7}，frequency为单位是MHz的整型数，voltage为浮点型的电压值。观察成员函数、默认构造函数、构造函数、拷贝构造函数和析构函数的调用情况及调用顺序。

```
#include <iostream>
using namespace std;

enum CPU_Rank{P1=1,P2,P3,P4,P5,P6,P7};  // 熟悉枚举类型定义方法，注意不要忘记最后的分号

class CPU{  // 类的声明，注意最后需要加分号
    CPU_Rank rank; // 默认是私有成员，直接放在类名的后面，关键privite可省
    int frequency;
    float voltage;
public:
    CPU(CPU_Rank r, int f, float v); // 声明构造函数，不能有返回值，在类外实现
    CPU(void){cout<<"正在调用CPU类的默认构造函数"<<endl;}  // 定义默认构造函数（隐式内联），形参void可省
    CPU(CPU &c);  // 声明拷贝构造函数，后面以显式内联函数实现
    ~CPU(){cout<<"正在调用CPU类的析构函数！"<< endl;} //定义析构函数（隐式内联），不能有返回值和形参（形参可加void）
    void run(void);  // 声明公有成员函数,类外实现
    void stop();
};

CPU::CPU(CPU_Rank r, int f, float v){  //构造函数在类外的实现
    rank = r;
    frequency = f;
    voltage = v;
    cout<<"正在调用CPU类的构造函数"<<endl;
}

inline CPU::CPU(CPU &c){  //显式内联函数实现拷贝构造函数
    rank = c.rank;
    frequency = c.frequency;
    voltage = c.voltage;
    cout<<"正在调用CPU类的拷贝构造函数"<<endl;
}
inline void CPU::run(){cout<<"CPU开始运行！"<<endl;}  //实现成员函数（显式内联）
void CPU::stop(){cout <<"CPU停止运行！"<<endl;} //实现成员函数

int main(){
```

```
    CPU a(P7, 300, 2.8); // 调用构造函数
    a.run(); //调用成员函数
    a.stop();
    CPU b; // 调用默认构造函数
    b.run();
    b.stop();
    CPU c(a);// 调用拷贝构造函数
    c.run();
    c.stop();
    CPU d = a; // 调用拷贝构造函数
    d.run();
    d.stop();

} // 逆序调用各个对象的析构函数
```

任务 2. 声明一个简单的 Computer 类，有数据成员芯片（cpu）、内存（ram）、光驱（cdrom）等，有两个公有成员函数 run、stop。cpu 为 CPU 类的一个对象，ram 为 RAM 类的一个对象，cdrom 为 CDROM 类的一个对象，声明并实现这个类。

```
#include <iostream>
using namespace std;

enum CPU_Rank{P1=1,P2,P3,P4,P5,P6,P7};
class CPU{ ...}; // 此处省略CPU类定义，详见任务一
class RAM{
    int ram;
public:
    RAM(int r){
        ram = r;
        cout << "调用了RAM的构造函数！"<<endl;
    }
    RAM(){cout<<"调用了RAM的默认构造函数！"<<endl;}
    RAM(RAM &r){
        ram = r.ram;
        cout << "调用了RAM的拷贝构造函数！"<<endl;
    }
    ~RAM(){cout<<"调用了RAM的析构函数！"<<endl;}
};
class CDROM{
    int cdrom;
public:
```

```
    CDROM(int c){
        cdrom = c;
        cout << "调用了CDROM的构造函数！"<<endl;
    }
    CDROM(){cout<<"调用了CDROM的默认构造函数！"<<endl;}
    CDROM(CDROM &c){
        cdrom = c.cdrom;
        cout<<"调用了CDROM的拷贝构造函数！"<<endl;
    }
    ~CDROM(){cout<<"调用了CDROM的析构函数！"<<endl;}
};

class Computer{
private:
    CPU cpu;
    RAM ram;
    CDROM cdrom;
public:
    Computer(CPU c, RAM r, CDROM cd){
        cpu = c; ram = r; cdrom = cd;
        cout<<"调用了Computer的构造函数！"<<endl;
    }
    Computer(){
        cout<<"调用了Computer的默认构造函数！"<<endl;
    }
    Computer(Computer &c){
        cpu = c.cpu; ram = c.ram; cdrom = c.cdrom;
        cout<<"调用了Computer的拷贝构造函数！"<<endl;
    }
    ~Computer(){cout<<"调用了Computer的析构函数！"<<endl;}
    void run(){cout<<"调用了Computer的run函数！"<<endl;}
    void stop(){cout<<"调用了Computer的stop函数！"<<endl;}

};

int main(){


    Computer A; // 调用Computer默认构造函数
    A.run(); //调用Computer成员函数
    A.stop();
    cout<<endl;

    CPU c(P5,300,2.8); // 调用CPU类构造函数，构造一个CPU c
```

```
    RAM r(4); //调用RAM类构造函数，构造一个G RAM r
    CDROM d; // 调用CDROM类默认构造函数，构造一个CDROM d
    Computer B(c,r,d); //调用Computer构造函数
    B.run();
    B.stop();
    cout<<endl;


    Computer C(A);// 调用拷贝构造函数
    C.run();
    C.stop();
    cout<<endl;


    Computer D = B; // 调用拷贝构造函数
    D.run();
    D.stop();
    cout<<endl;


} // 逆序调用各个对象的析构函数
```